Midterm

(75 minutes open book exam)

	credit	max
Question 1		20
Question 2		20
Question 3		20
Question 4		20
Question 5		20
Total		100

Name: ______________________________

Name: ______________________________

Question 1. ($20 = 10 + 5 + 5$ points). Consider a set of n intervals of the form $[a_i, b_i]$ with $0 \le a_i \le b_i \le 1$ for each $1 \le i \le n$.

(a) Describe an algorithm that finds a maximum number of disjoint intervals in the set. Justify your solution.

(b) Analyze the running time of your algorithm.

(c) You get the last five points if your algorithm runs in time O($n \log n$).

Solution.

(a) At each step we get the interval in the remaining set which minimum right endpoint. Then we remove the intervals that intersect the one just chosen and we repeat. For the implementation, we store the intervals in a priority queue ordered by right endpoint. The intervals are removed implicitly as we add them to the set only if they are disjoint from the intervals that are already there.

```
$R = -\infty$;
for $i = 1$ to $n$ do
  $[a, b]$ = MINEXTRACT;
  if $R < a$ then
    add $[a, b]$ to the set;
    $R = b$
  endif
endfor.
```

(b) The running time is O(n) for constructing the priority queue and O($\log n$) to remove each intervals. The total is O($n \log n$).

(c) As shown in (b), the running time is O($n \log n$).

Name: ______________________________

Question 2. ($20 = 7 + 13$ points). Suppose you count from 0 to n, storing the digits of the current number, j, in ternary notation in a linear array, that is,

$$j = \sum_{i \geq 0} A[i] \cdot 3^i,$$

where each $A[i]$ is either 0, 1, or 2.

(a) Describe the algorithm that does the counting.

(b) If you charge $1 each time the algorithm changes a digit, what is the amortized cost per increment?

Solution.

(a) We assume $A[i] = 0$ for all i when we start the algorithm.

```
for $j = 0$ to $n$ do
  $i = 0$;
  while $A[i] = 2$ do
    $A[i] = 0$;  $i = i + 1$
  endwhile;
  A[i] = A[i]+1
endfor.
```

(b) Whenever we change a 0 to a 1, we add $1.5 to the system. Of this, we take $1 to pay for the change and leave $0.5 with the digit. Similarly, whenever we change a 1 to a 2 we add $1.5 to the system, again taking $1 to pay for the change and leaving $0.5 with the digit. Together with the earlier $0.5 this adds to $1. Whenever we change a 2 to a 0, we use this $1 to pay for the change. Each time we increment j, we have some changes of the last type and exactly one change of either the first or the second type. Hence, we add a total of $3n/2$ to the system proving that this is an upper bound on the total cost.

Name: ______________________________

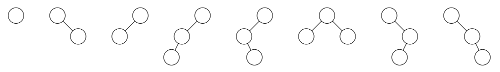
Question 3. ($20 = 7 + 13$ points). Construct a binary search tree by successively inserting n distinct items into an initially empty tree, without ever rebalancing the tree. Let $M(n)$ be the number of different trees you can get.

(a) Draw the $M(n)$ trees of n nodes for $n = 0, 1, 2, 3$.

(b) Write a recurrence relation that expresses $M(n)$ in terms of $M(0)$ to $M(n-1)$.

Solution.

(a) We have $N(0) = 1$, $N(1) = 1$, $N(2) = 2$, $N(3) = 5$, with trees shown in the figure below.



(b) The root partitions the remaining $n-1$ nodes into i nodes to the left and $n-i-1$ nodes to the right. For each such partition, the number of possibilities is the number of possible left subtrees times the number of possible right subtrees. More formally,

$$M(n) = \sum_{i=0}^{n-1} M(i) \cdot M(n-i-1).$$

Just to check the formula, we get $M(3) = M(0)M(2) + M(1)M(1) + M(2)M(0) = 2 + 1 + 2 = 5$.

Name: ______________________________

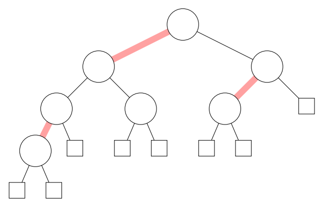
Question 4. $(20 = 8 + 12$ points). Recall that a binary search tree is an AVL tree if for every node the height difference between its left subtree and its right subtree is at most 1.

(a) Draw the smallest AVL tree of height three (height four if you count the leaves) and color the edges red and black so as to show that this tree has the structure of a red-black tree.

(b) Prove that every AVL tree has the structure of a red-black tree. In other words, show that the edges can be colored red and black in a way that satisfies the conditions of a red-black tree.

Solution.

(a) The AVL of height four (counting the edges to the leaves) has seven internal nodes and is shown in the figure below. Starting from the leaves we color the edges first black and then red whenever this is necessary to guarantee the same black-depth for all leaves.



(b) Let h be the height of the AVL tree, A. We construct a red-black tree of the same shape starting with the complete binary tree, A', of height h. To begin, we color the edges of each path from the root to a leaf in A' alternating black and red such that the last edge is black. Let v be the root of the AVL tree, u its left child, and w its right child. Similarly, let v' be the root of A', u' its left child and w' its right child. If the tree rooted at u is higher than that rooted at w then remove one subtree of w' and contract the two edges going in and out of w' by a single, black edge, thus removing w'. Note that this operation preserves the red-black tree properties because one of the two contracted edges is red and the other is black. Do the symmetric operation if the tree rooted at w is higher than that rooted at u, and retain the structure at the root of A' if the trees rooted at u and w have the same height. Now recurse for the left and right subtrees of A and of A'.

The change replaces a red and a black edge by one black edge. It thus preserves the red-black tree property of A'. Furthermore, it brings the structure of A' one step closer to the structure of A. After a finite number of steps, A' has the same structure as A.

Name: ______________________________

Question 5. (20 points). Given a heap of n items, each a real number, the operation ListMin(x) enumerates all items smaller than x. Give an algorithm that implements ListMin and takes time proportional to the number of items enumerated.

Solution. The algorithm is recursive, starting at the root of the heap. If x is larger than the item at the root then we output the item and we call the algorithm for the left child as well as for the right child of the root. If we enumerate k items, the algorithm examines at most $2k + 1$ items. This is less than $3k$ unless $k = 0$.